Remarks

The amendments to claims 2 and 5

The amendments to claims 2 and 5 have been made to make it clear that the snapshot of claims 2 and 5 is not a simple copy of the first configuration tables. The amendments are supported by the description of the snapshot table `Config_snapshot` at page 7, lines 16-19 of Applicants' Specification as filed.

Traversal of the rejections

As set forth in MPEP 2142, an examiner who rejects a claim under 35 U.S.C. 103 must establish a *prima facie* case of obviousness. Part of the requirements for the *prima facie* case of obviousness is that

> the prior art reference (or references when combined) must teach or suggest all the claim limitations. (MPEP rev. 3, August 2005, §2142, page 2100-136, col. 1)

Applicants will show in the following that the references do not teach or suggest all of the claim limitations and that Examiner has consequently not made his *prima facie* case of obviousness.

In general

There are many situations in digital systems where it is useful to determine whether something in the system has "changed underfoot" before undertaking an action. Migrating from one system configuration to another as described in Applicants' Specification and set forth in Applicants' claims is one such situation. The references cited thus far in the prosecution disclose other such situations.

Gere and Dennis

In the cases of Gere and Dennis, the situation is booting up a system and what needs to be determined is whether the boot record has changed. That is done by comparing the copy of the boot record which will be used to boot the system with a previously-made copy.

Anonsen

In the case of Anonsen, two such situations are disclosed: one is optimistic concurrency in database systems. The situation there is the following: in database systems, a common activity is performing a query to read a current value of a field of a record in the database system, performing an operation based on the read value, and then writing to one or more fields in the database system. If the field of the record being read is not locked, another user may give the field a new current value while the operation based on the read value is being performed. The change in the current value renders the operation based on the read value invalid; consequently, the database system must check whether the current value has changed before writing the fields in the database system. As described at par. 0016 of Anonsen, the checking is generally done by comparing the read value and the current value of the record field or by determining whether the row has been updated since the read value was read.

The other situation is concurrency in object oriented databases. See in this regard Anonsen's FIG. 1 and the description beginning at paragraph 0028. In these databases, classes of objects are defined and the class definition of an object maps attributes of the object onto columns of one or more tables in the database system. An instance of an object is made by performing a query that fetches the values of the instance's attributes from the database system. An operation may then be performed on the instance and changed attribute values may be written back to the database system. If there is concurrent access to the attributes of the object in the database system, the problem described above for database fields exists here as well, and checking must be done to be sure that the values of the object's attributes have not changed underfoot. As described at paragraph 0045, the checking is done by saving the original values of the attributes and comparing them with the current values of the fields they were obtained from before writing the changed attribute values back to the database system.

As is apparent from the foregoing, the general technique of determining whether something has "changed underfoot" before performing an action may be implemented in

many ways, depending on the context in which the determination is being made. Dennis and Gere do not determine whether a boot record has changed underfoot in the same way that Anonsen determines whether a field in a record in a database has changed underfoot and Applicant determines whether a change has occurred in a set of configuration tables in a different way entirely. Applicants are not claiming the general technique of determining whether something has "changed underfoot", but rather their technique for doing so in the context of a system whose configuration is defined by configuration tables in a database.

What Applicants are claiming

Claims 1 and 26

Applicants' claim 1 sets forth in straightforward fashion what Applicants are claiming:

> **1.** A method of migrating from configuration *m* of a system to a configuration *m*+1 thereof, the system's configuration being defined by first configuration tables in a database and
> the method comprising the steps performed by the system of:
> making second configuration tables that define configuration *m*+1;
> making a determination whether the first configuration tables still define configuration *m;* and
> if the first configuration tables still define configuration *m*, using the second configuration tables to modify the first configuration tables such that the first configuration tables define configuration *m*+1.

In Applicants' claim 1, the configuration of the system is defined by *configuration tables in a database.* A change from a configuration *m* of the system which is defined by first configuration tables to a configuration *m*+1 which is defined by second configuration tables is made by making the second configuration tables and using the second configuration tables to modify the first configuration tables such that the first configuration tables define configuration *m*+1. As Examiner points out, changing the system configuration by making a copy of the system, changing the configuration of the copy by changing the copy's configuration tables, and then changing the original configuration tables so that they agree with the copy's configuration tables is disclosed in AAPA.

The issue between Examiner and Applicants is whether Anonsen discloses the limitations set forth at lines 6-10 of the claim. The limitations require two sets of configuration tables, one defining a configuration *m* of a system and the other defining a configuration *m*+1. Before modifying the first set of configuration tables so that they define configuration *m*+1, a determination is made whether the first set of configuration tables still defines configuration *m*. If they do, the first set of configuration tables is modified so that they define configuration *m*+1. The kinds of tables included in a preferred embodiment's configuration tables 129 are described at page 3, lines 15-21:

> The tables in configuration tables 129 fall into four groups: state machine tables, which define what activities system 101 performs and how the performance of the activity affects the state of system 101, permission tables, which define the permissions held by various users of system 101, notification tables, which define who is to be notified and how when an activity is performed, login tables, which define how users must login, and name definition tables, which define the names used for entities in system 101.

Details of the configuration tables are not necessary in the context of the present application, and so they are not described further. It will, however, be immediately apparent to those skilled in the art that the set of configuration tables for a system like that in which a preferred embodiment of the present invention is implemented includes a large number of tables and that many of these tables will themselves be large. Should further confirmation of that fact be necessary, the configuration tables are described in more detail in USSN 10/438,581, Flam, et al., *Techniques for providing audit trails of configuration changes*, which is incorporated by reference into the present patent application. A copy of FIG. 1 of 10/438,581 showing the tables making up configuration tables 189 in a preferred embodiment is included with this response. Clearly, "making a determination whether the first configuration tables still define configuration *m*" is a very different matter from determining whether two copies of a boot record are identical or determining whether a field in a particular database record has changed its value. Because that is the case, the combination of AAPA and Anonsen does not disclose all of the limitations of Applicants' claim 1 and Examiner has not made his *prima facie* case of obviousness. Claim 26 incorporates all of the limitations of claim 1 and is therefore patentable over the references for the same reasons as claim 1.

Rebuttal of Examiner's rejection of claim 1

Examiner's failure to make a *prima facie* case is confirmed when his rejection is looked at in detail. The pertinent part of the rejection, beginning at page 3, line 6 reads as follows:

> Anonsen teaches another method of transitioning from an original table to an updated table while also checking for changes so as not to erase important changes. Anonsen teaches a method of creating a copy of an initial table, then updating the table; however before the table is updated with the new updated table the initial table is compare with the copy to determine if the tables match wherein if the tables do not match it is determined that a change has occurred and if the tables do match no change has occurred and the table is updated.

In making this rejection, Examiner refers Applicants to Anonsen's FIG. 3 and the discussion at paragraphs 0044-0046. As already set forth above and as is clear from FIG. 3 and the cited paragraphs, the determination made by Anonsen is whether the database record fields onto which the attribute values of an instance of an object are mapped have changed between the time the object instance was created and the time the object's current attribute values are to be written back to the record fields. Anonsen makes the determination by saving a copy of the object's attribute values at the time the object is created and comparing the attribute values in the copy with the values of the record fields to determine whether the values of the record fields have changed. See in this regard boxes 1014, 1016, and 1018 of FIG. 3 and the cited locations. Anonsen thus does not compare entire tables, but only fields of records in the tables and consequently discloses nothing whatever about "transitioning from an original table to an updated table while checking for changes" or about "creating a copy of an initial table, then updating the table; however before the table is updated with the new updated table the initial table is compared with the copy to determine if the tables match …". Since that is the case, Anonsen does not supply the limitation lacking in AAPA.

Claim 19

Claim 19 sets forth that a snapshot table is used to detect whether the first configuration tables still define configuration *m*. Anonsen's copies of the values of the attributes from the fields onto which the instance of the object is mapped are not a snapshot table; consequently, the combination of AAPA and Anonsen also does show all of the limitations of claim 19 and the claim is patentable over the references.

The dependent claims

The dependent claims are of course all patentable because they all inherit limitations that are not disclosed in the combination of AAPA and Anonsen; However many of the dependent claims have additional limitations that are not disclosed in the references, and consequently, the dependent claims are patentable over the references.

Claims 2, 3, 5, and 6

In Anonsen and also in Gere and Dennis, the "snapshot" is a copy of what is being snapshotted; the amendment to claims 2 and 5 make it clear that Applicants' snapshot is not a copy. The references consequently do not disclose the "snapshot" limitation of claims 2, 3, 5, and 6.

Claims 8-10, 12-16, 20, 23, and 24

These claims are also patentable in their own rights over the references. As regards claims 8-9, 13-16, and 23-24, there is no disclosure in either the AAPA or the references of the arrangements for ensuring that users have been logged off before the snapshot is made or that the users who must approve configuration *m*+1 must do so. The added limitations of these claims are thus not disclosed in the references and are therefore patentable in their own rights over the references. Claims 10, 12, and 20 all describe the interaction between the user and the system of claim 1 "when the first configuration tables no longer define configuration *m*", namely that the user is notified and if the user so indicates, the system "overwrite[s] the first configuration tables with the second configuration tables".

Conclusion

Applicants have shown that their amendments to claims 2 and 5 are fully supported by the claims and have traversed all of Examiner's rejections and have therefore been completely responsive to Examiner's Office action as required by 37 C.F.R. 1.111(b). Applicants therefore respectably request that Examiner reconsider his rejections as provided by 37 C.F.R. 1.111(a) and allow the claims. No fees are believed to be required for this response. Should any be, please charge them to deposit account number 501315.

Respectfully submitted,

/Gordon E. Nelson/
Attorney of record,
Gordon E. Nelson
57 Central St., P.O. Box 782
Rowley, MA, 01969,
Registration number 30,093
Voice: (978) 948-7632
Fax: (866)-723-0359
10/26/06
Date